|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COGNOME:** |  | **NOME** |  | **B3** |
| **MATRICOLA:** |  | | |
| **DOCENTE:** |  | | |

|  |  |
| --- | --- |
| **Domanda 1** | *Risultato* |
| Dati i seguenti numeri espressi in decimale, si convertano in binario CA2 su 8 bit, se ne effettui la somma e si determini l’esistenza della condizione di overflow.  A=-3, B=19  C=-30, D=41 | *Risultato*  A=\_\_\_\_\_\_\_ A+B=\_\_\_\_\_\_\_\_  B=\_\_\_\_\_\_\_ overflow? \_\_\_\_\_  C=\_\_\_\_\_\_\_ C+D=\_\_\_\_\_\_\_\_  D=\_\_\_\_\_\_\_ overflow? \_\_\_\_\_ |
| *Passaggi più significativi per arrivare al risultato* | |

|  |
| --- |
| **Domanda 2** |
| Date le due funzioni booleane: f(A,B,C) = ((A and B) or C) e g(A,B,C) = ((A or (B xor C)) determinare per quali valori di A, B e C producono risultati equivalenti. |
|  |

|  |
| --- |
| **Domanda 3** |
| Descrivere brevemente come vengono codificati i caratteri in un elaboratore. |
|  |

|  |
| --- |
| **Domanda 4 (PROGRAMMAZIONE)** |

Si realizzi un programma strutturato in linguaggio C per gestire i costi di un parcheggio auto sotterraneo aperto dalle 7.00 del mattino alla mezzanotte di ogni giorno. I prezzi orari del parcheggio sono definiti secondo diverse fasce orarie memorizzate in un file chiamato “prezzi.txt” che contiene le fasce orarie e i prezzi codificati nel seguente formato:

<ORA1>:<MINUTI1> <ORA2>:<MINUTI2> <PREZZO>

dove:

* le ore e i minuti indicano l’inizio e la fine della fascia oraria
* il prezzo indica il costo in euro per ogni 5 minuti di parcheggio

Un secondo file, il cui nome è passato come primo parametro da linea di comando, contiene le targhe dei veicoli che hanno acceduto al parcheggio nell’arco di una giornata codificate nel seguente formato:

<TARGA> <ORA1>:<MIN1> <TEMPO>

dove:

* le targhe sono codificate da 7 caratteri
* ORA1 e MIN1 rappresentano l’orario di accesso al parcheggio
* TEMPO rappresenta il numero di minuti di utilizzo del parcheggio. Considerare il caso in cui il veicolo non sia uscito dal parcheggio: in questo caso l’orario di uscita riporta il carattere X.

Si considerino le ore e i minuti espressi come numero intero mentre il prezzo come numero reale. Il file “prezzi.txt” ha esclusivamente 4 righe mentre la lunghezza del file degli accessi al parcheggio non è noto a priori. **Il prezzo di una sosta è calcolato esclusivamente sulla base del prezzo all’ora di INGRESSO come minuti\_sosta \* prezzo\_fascia / 5.**

Ad esempio un veicolo che accede alle 11.28 ed esce alle 11.32 pagherà 0.2 €.

Il programma, ricevuto in ingresso come secondo parametro da linea di comando il codice di targa di un veicolo, deve:

* verificare che il veicolo sia presente nel file
* in caso affermativo
  + indicare se il veicolo ha effettuato la sosta più corta nel parcheggio nel caso in cui il veicolo sia uscito almeno una volta dal parcheggio
  + indicare se il veicolo è ancora all’interno del parcheggio a fine giornata
  + stampare il prezzo totale pagato dal veicolo nel caso in cui il veicolo sia uscito almeno una volta dal parcheggio.

Esempi file:

|  |  |
| --- | --- |
| *Accessi\_20160204.txt* | *Prezzi.txt* |
| POP4COU 08:32 68  JKE4437 11:20 456  BBQSAU5 15:40 130  POP4COU 19:10 80  TOP3GUN 23:40 X  BBQSAU5 23:50 X | 07:00 11:30 0.25  11:30 15:30 0.20  15:30 20:00 0.15  20:00 23:59 0.10 |

Risultati di esecuzione:

|  |
| --- |
| C:\> prog.exe Accessi\_20160204.txt JKE4437  Veicolo presente. Non ha effettuato la sosta più corta.  Prezzo pagato: 22.80 €  Il veicolo è uscito dal parcheggio. |

|  |
| --- |
| C:\> prog.exe Accessi\_20160204.txt POP4COU  Veicolo presente. Ha effettuato la sosta più corta  Prezzo pagato: 2.4 €  Il veicolo è uscito dal parcheggio. |

|  |
| --- |
| C:\> prog.exe Accessi\_20160204.txt GIASTIT  Veicolo non presente. |

|  |
| --- |
| C:\> prog.exe Accessi\_20160204.txt TOP3GUN  Veicolo presente.  Il veicolo non è uscito dal parcheggio. |

|  |
| --- |
| C:\> prog.exe Accessi\_20160204.txt BBQSAU5  Veicolo presente. Non ha effettuato la sosta più corta  Prezzo pagato: 3.90 €  Il veicolo non è uscito dal parcheggio. |

**#include <stdio.h>**

**FILE \*fopen(char \*filename, char \* mode**) – Apertura di un file (mode: “r” lettura – “w” scrittura – “a” append)

**FILE \*freopen(char \*filename, char \* mode, FILE \*file\_pointer**) - Riassegna un file puntatore ad un file diverso.

**int fclose(FILE \*file\_pointer)** - Chiude un file

**int feof(FILE \*file\_pointer) -** Controlla se e' stato incontrato un end-of-file in un file.

**int fflush(FILE \*file\_pointer) -** Svuota il buffer di un file.

**int getchar(void) -** Legge un carattere da "stdin" (tastiera)

**int fgetc(FILE \*file\_pointer) -** Prende un carattere da un file

**char \*gets(char \*buffer) -** Legge una riga da "stdin" (tastiera)

**char \*fgets(char \*string, int maxchar, FILE \*file\_pointer) -** Legge una riga da un file.

**int printf(char \*format \_string, …) -** Scrive output formattato su "stdout" (schermo)

**int fprintf(FILE \*file\_pointer, char \*format\_string, …) -** Scrive output formattato in un file.

**int sprintf(char \*string, char \*format\_string, …) -** Scrive output formattato su una stringa

**int fputc(int c, FILE \*file\_pointer) -** Scrive un carattere in un file

**int putchar(int c) -** Scrive un carattere su "stdout" (schermo)

**int puts(char \*string)** - Scrive una stringa su "stdout" (schermo)

**int fputs(char \*string, FILE \*file\_pointer)** - Scrive una stringa in un file.

**int scanf(char \*format\_string, args)** - Legge input formattato da "stdin" (tastiera)

**int fscanf(FILE \*file\_pointer, char \*format string, args) -** Legge input formattato da file

**int sscanf(char \*buffer, char \*format\_string, args) -** Legge input formattato da una stringa

**EOF** – end of file (costante a valore negativo)

**NULL** - puntatore nullo (valore 0)

**#include <stdlib.h>**

**double atof(char \*string)** - Converte una stringa in un valore in floating point.

**int atoi(char \*string)** - Converte una stringa in un valore integer.

**int atol(char \*string)** - Converte una stringa in un valore long integer.

**void exit(int val)** – Termina il programma, restituendo il valore ‘val’.

**EXIT\_FAILURE** - costante per segnalare terminazione senza successo del programma con exit(); valore diverso da zero

**EXIT\_SUCCESS** - segnala terminazione con successo del programma con exit(); vale 0

**#include <string.h>**

**char \*stpcpy (char \*dest, char \*src) -** Copia una stringa in un'altra. Restituisce dest

**char \*strncpy(char \*s1, char \*s2, size\_t n) -** Copia i primi "n" caratteri di s2 in s1. Restituisce s1

**int strcmp(char \*s1, char \*s2) -** Confronta s1 e s2 per determinare l'ordine alfabetico (<0, s1 prima di s2, 0 uguali, >0 s1 dopo s2)

**int strncmp(char \*s1, char \*s2, size\_t n) -** Confronta i primi "n" caratteri di due stringhe.

**char \*strcpy(char \*s1, char \*s2) -** Copia s2 in s1. Restituisce s1

**int strlen(char \*string) -** Determina la lunghezza di una stringa.

**char \*strcat(char \*s1, char \*s2, size\_t n) -** Aggiunge s2 a s1. Ritorna s1

**char \*strncat(char \*s1, char \*s2, size\_t n) -** Aggiunge "n" caratteri di s2 a s1. Ritorna s1

**char \*strchr(char \*string, int c) -** Cerca la prima occorrenza del carattere ‘c’ in string; restituisce un puntatore alla prima occorrenza di c in s, NULL se non presente

**char \*strrchr(char \*string, int c) -** Cerca l'ultima occorrenza del carattere ‘c’ in string

**char\* strstr(char\* s, char\* t)** - Restituisce un puntatore alla prima occorrenza di t all'interno di s. Restituisce NULL se t non è presente in s.

**char\* strtok(char\* s, const char\* t) -** scompone s in token, i caratteri che delimitano i token sono contenuti in t. Restituisce il puntatore al token (NULL se non ne trova nessuno). Alla prima chiamata in s va inserita la stringa da scomporre e in t i caratteri che delimitano i vari token. Per operare sulla stessa stringa, alle successive chiamate al posto di s si deve passare NULL

**#include <ctype.h>**

**int isalnum(int c)** - Vero se ‘c’ e' alfanumerico.

**int isalpha(int c)** - Vero se ‘c’ e' una lettera dell'alfabeto.

**int iscntrl(int c)** - Vero se ‘c’ e' un carattere di controllo.

**int isdigit(int c)** - Vero se ‘c’ e' un numero decimale.

**int islower(int c)** - Vero se ‘c’ e' una lettera minuscola.

**int isprint(int c)** - Vero se ‘c’ e' un carattere stampabile.

**int ispunct (int c)** - Vero se ‘c’ e' un carattere di punteggiatura.

**int isspace(int c)** - Vero se ‘c’ e' un carattere spazio.

**int isupper(int c)** - Vero se ‘c’ e' una lettera maiuscola.

**tolower(int c)** - Converte ‘c’ in minuscolo.

**int toupper(int c)** - Converte ‘c’ in maiuscolo.

**#include <math.h>**

**int abs (int n) –** valore assoluto intero

**long labs(long n) –** valore assoluto long

**double fabs (double x ) –** valore assoluto di x

**double acos(double x) -** arcocoseno

**double asin(double x) -** arcoseno

**double atan(double x) -** arcotangente

**double atan2(double y, double x) –** arcotangente di y/x.

**double ceil(double x) –** intero superiore a x

**double floor(double x) –** intero inferiore a x.

**double cos(double x) –** x in radianti

**double sin(double x) –** x in radianti

**double tan(double x) –** x in radianti

**double cosh(double x) –** coseno iperbolico

**double sinh(double x) –** seno iperbolico

**double tanh(double x) –** tangente iperbolica

**double exp(double x) -** ex

**double log(double x) -** log(x).

**double log10 (double x ) –** logaritmo base 10

**double pow (double x, double y) -** xy

**int rand (void) –** intero casuale tra 0 e RND\_MAX.

**int random(int max\_num) –** valore casuale tra 0 e max\_num.

**void srand(unsigned seed) –** inizializza la sequenza di valori casuali

**double sqrt(double x) –** radice quadrata

**#include <limits.h>**

**INT\_MAX -** Indica il più grande valore che è possibile rappresentare con un int.

**INT\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un int.

**LONG\_MAX -** Indica il più grande valore che è possibile rappresentare con un long.

**LONG\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un long.

**#include <float.h>**

**FLT\_MAX, DBL\_MAX -** Indica il più grande valore che è possibile rappresentare con un float (o double)

**FLT\_MIN, DBL\_MIN -** Indica il più piccolo valore che è possibile rappresentare con un float (o double)